Object detection system based on SSD algorithm

Qianjun Shuai, Xingwen Wu
School of Information and Communication Engineering
Communication University of China, CUC
Beijing, China
e-mail: sqj, xingwen@cuc.edu.cn

***Abstract*—SSD (Single Shot Multi Box Detector) is an object detection algorithm based on deep learning. As one of the most mainstream detection algorithms, it can greatly improve the detection speed and ensure the detection accuracy. In this paper, the Batch Norm operation is added to the network in order to improve the generalization of the network and speed up network training. The object counting function is added to the image recognition. This paper uses SSD algorithm that incorporates Batch Norm algorithm. The object detection system was built by the Flask framework and the Layui framework. The system can select the data to be detected on the front-end page, the detection results and the number of each type of object were displayed on the front-end page in real time.**

Keywords-SSD algorithm; Batch Norm algorithm; Object counting; Flask.

I. INTRODUCTION

Object detection is an extremely important and very hot topic in the current computer vision field. The purpose of object detection is to identify the object in the picture and use the bounding box to locate the object. With the development of deep learning and the needs of the monitoring field, the object detection technology has made great progress. That has been widely used in the fields of intelligent monitoring, traffic management, and security. At present, Object detection algorithms based on deep learning are divided into two categories: models based on regional candidate boxes and models based on regression.

The deep learning object detection model based on regional candidate boxes is divided into two stages: The first stage is the selection of candidate boxes, the second stage is the extraction and classification of features. Typical representatives of such models are R-CNN [1], SPP-net [2], Fast R-CNN [3], Faster R-CNN [4], R-FCN [5], etc. Although the algorithms represented by Faster R-CNN have achieved end-to-end training. However, these algorithms are difficult to detect in real time due to the complex network structure and many training parameters. The regression-based deep learning object detection model requires pre-drawing default boxes in a certain way, and then classifying objects and predicting borders on each default box. Typical representatives of such models are YOLO [6] and SSD [7]. The idea of YOLO is to divide the image into grids of S×S size, and then predict the bounding box and category of each grid. It discards the suggestions of candidate regions, makes the network simpler, and greatly improves the detection speed. However, The YOLO model also limits the model's ability to predict nearby objects, thus leading to missed detection of dense targets. The object detection accuracy of the YOLO model was also reduced by 10% compared to Fast R-CNN [8]. The SSD model combines the advantages of both Faster R-CNN and YOLO. Based on YOLO, The SSD model takes advantage of the idea of RPN to ensure high inspection accuracy and speed at the same time. Later, On the basis of SSD, Fu et al. [9] replaced the original VGG16 [10] with the residual neural network Reisidual-101 [11] that has stronger feature extraction capabilities. The de-convolution module is also introduced in the network. Finally, Fu et al proposed the new DSSD model. Although the DSSD model improves the detection accuracy, there is a significant decrease in the detection speed. The R-SSD model proposed by Jeong et al. [12] has the same problem.

In this paper, the SSD network structure is modified, and The Batch Norm operation is added before the feature fusion layer to accelerate network training. The target counting function is added to the object detection. Finally, the usable object detection system was built through the Flask framework and the Layui framework.

II. SSD ALGORITHM

A. Introduction to the SSD algorithm

The SSD model is a one-stage object detection network. It draws on the anchors mechanism in Faster R-CNN, combines the idea of YOLO regression, and expresses the characteristics of different scale features. The SSD model adopts a multi-scale target feature extraction method, which makes the detection speed of SSD model faster than Faster R-CNN and the detection accuracy is higher than YOLO.

The SSD model can be divided into the following parts: the backbone network part, the original bounding box generation part and the convolution prediction part. The backbone network part could further divided into the basic network and the additional feature extraction layer. The convolution prediction includes object category prediction and position prediction. The main process of the algorithm is as follows: First, the images are fed into the network to extract features using deep neural networks. Second, design different default boxes used to extract feature maps at different scales. Third, the features in the default frames are extracted to predict the type and location of the target. Finally, the non-maximal suppression algorithm (NMS) is

978-1-7281-8138-7/20/$31.00 ©2020 IEEE
DOI 10.1109/ICCST50977.2020.00033

used to select the prediction result that best matches the real target box.

B. SSD network structure

The basic network of the SSD uses the VGG16 network. The two fully connected layers FC6 and FC7 of the VGG16 are replaced by convolution layers. All Dropout layers and FC8 layers are removed. Four convolution layers Conv8_2, Conv9_2, Conv10_2, and Conv11_2 are added after the modified VGG16. The new convolution layer is used to obtain more features for detection. The SSD network structure as shown in Fig. 1

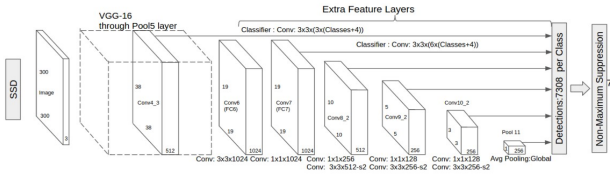

Figure 1. SSD network structure diagram

C. Identify candidate boxes

The SSD model borrows the concept of anchor from Faster R-CNN, where each cell sets priori frames with different aspect ratios. The SSD model uses a pyramid network structure to comprehensively extract the feature maps of the Conv4_3, Conv7, Conv8_2, Conv9_2, Conv10_2, and Conv11_2 layers. Each point on top of these feature map layers constructs six candidate boxes of different scale sizes, and then combines the candidate boxes obtained from different feature maps. Non-maximum suppression (NMS) method is used to filter out some of the overlapping or incorrect candidate boxes to produce the final set of candidate boxes. Since the SSD model not only obtains feature mappings at different scales, but also makes predictions on top of different feature mappings. The SSD model is able to accurately detect objects at different scales. Here are the rules for generating six candidate boxes of different scale sizes for each point.

- Centered on the midpoint of each point on the feature map (offset=0.5) generates a series of concentric Default boxes.
- Using m feature maps of different sizes to make predictions. The scale of the underlying feature map, the value is set to min = 0.2. The scale of the top feature map, the value is set to max = 0.95. The other layers are calculated by (1).

$$s_k = s_{\min} + \frac{s_{\max} - s_{\min}}{m-1}(k-1), \quad k \in [1, m] \tag{1}$$

- Using different ratio values, a_r = (1, 2, 3, 1/2, 1/3), calculate the width (w) and height (h) of the default box via (2) and (3).

$$w_k^a = s_k \sqrt{a_r} \tag{2}$$

$$h_k^a = s_k / \sqrt{a_r} \tag{3}$$

- For ratios = 0, the specified scale is calculated according to (4).

$$s_k' = \sqrt{s_k s_{k+1}} \tag{4}$$

D. Loss function

The loss function of the SSD algorithm is designed based on the output of the prediction part. The loss function is the sum of the confidence loss and the position loss. The formula for the loss function is shown in (5).

$$L(z,c,l,g) = \frac{1}{N}\left(L_{conf}(z,c) + \alpha L_{loc}(z,l,g)\right) \tag{5}$$

Where N is the number of prediction boxes that match the ground truth object box, $L_{conf}(z,c)$ represents the confidence loss, $L_{loc}(z,l,g)$ represents the position loss, z indicates whether the prediction box matches the ground truth target box, if it does z is equal to 1, otherwise it is 0, c indicates the confidence of the prediction box, l indicates the information on the location of the prediction box, g denotes the location information of the ground truth object box, α is weight coefficient, which is used to determine the weight relationship between the confidence loss and position loss. Generally, the two losses take the same weight, and the value is set to 1.

III. MODIFY THE MODEL

A. Batch Norm algorithm

Batch Norm algorithm was designed to solve the problem of data distribution during training, which improves network generalization and speeds up network training. The essence of Batch Norm algorithm is to pre-process the data and normalize it before feeding the data into the network, which reduces variation in the data distribution and makes the network much more generalizable and faster to train.

Data preprocessing is often done in neural networks with a whitening operation, but the operation is too computationally intensive and not microscopic everywhere. So the Batch Norm algorithm improves on the whitening operation. Each dimensional feature is independently normalized to a vector with a mean of 0 and a variance of 1. But if the Batch Norm algorithm is simply normalized in this way, it will affect the features learned by the network layer. To solve this problem, two parameters γ and β are introduced to deflate and shift the normalized values. The Batch Norm algorithm preprocessing process is shown in (6)-(9).

$$\mu_B \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \tag{6}$$

$$\sigma_B^2 \leftarrow \frac{1}{m}\sum_{i=1}^{m}\left(x_i - \mu_B\right)^2 \tag{7}$$

$$\hat{x}_i \leftarrow \frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + \varepsilon}} \tag{8}$$

$$y_i \leftarrow \gamma \hat{x}_i + \beta \tag{9}$$

Equations (6)-(9) represent the mean, variance, normalization, and reconstruction transformations of small batches. ε is constant to ensure numerical stability of the small batch variance.

In this paper, the Batch Norm operation is added before the Conv4_3, Conv7, Conv8_2, Conv9_2, Conv10_2, and Conv11_2 layers. Then the output feature map is fused with features and sent to the prediction network for prediction computation.

B. Object counting function

In this paper, the Pascal VOC dataset is used to train the SSD network. There are 20 types of objects in the Pascal VOC dataset, so when counting the objects, first create a dictionary with the object name as the key and all values set to 0. In the process of object detection, the number of final candidate boxes is summed up by labels and stored in the dictionary.

IV. Experiments and Analysis of Results

A. Experimental data

The dataset used in this paper is the Pascal VOC, which is a standardized set of object detection datasets. The data set includes 20 categories. The composition of the data set is shown in TABLE I.

TABLE I. Components of the Pascal VOC dataset

Dataset	trainval	test
Pascal VOC2007	5011	4952
Pascal VOC2012	17125	5138

The modified SSD model was trained using the VOC2007trainval dataset and VOC2012trainval dataset. The model was tested by the VOC2007test dataset and VOC2012test dataset.

B. Evaluation index

In this experiment, mAP (mean average precision) was used as the evaluation index. mAP takes into account both precision and recall, so it is often used as a model evaluation index for multi-target detection. The formula for mAP is given in (10).

$$mAP = \frac{\sum_{q=1}^{Q} AveP(q)}{Q} \tag{10}$$

In the formula, Q is the type of objects, q refers to the detection rate of a type of object at different recall rates, $Avep(q)$ refers to the average precision of a class of object.

C. Experimental and Analysis of results

In network training, the size of each batch was 16, the initial learning rate was 0.01, and the decay factor of the learning rate was 0.94. The model was saved once every 1000 iterations, the maximum number of iterations was 50000. The test results in the VOC2007 test set were shown in TABLE II. The test results in the VOC2012 test set were shown in TABLE III.

TABLE II. VOC2007 test set test results comparison

Model	mAP/%	FPS
SSD	76.2	46
SSD+BN	78.1	38

TABLE III. VOC2012 test set test results comparison

Model	mAP/%	FPS
SSD	77.8	46
SSD+BN	81.2	38

The experimental results in TABLE II and TABLE III show that the modified SSD model has a good performance in the VOC2007 test set and VOC2012 test set, the detection accuracy has been improved compared to the original SSD model. However, the detection speed of the modified model has decreased slightly to 38 FPS. Comprehensive comparison of the model's detection accuracy and detection speed, the model that incorporates Batch Norm operation has improved performance relative to the original SSD.

V. Visualization of Object Detection Systems

The object detection system was built by the Flask framework and Layui framework Through CSS for simple rendering of the page, while using JS and Ajax to establish a link between the front and back, so that the back-end application to respond to front-end requests and data transfer to the front-end to display.

A. Picture detection visualization

In the front-end page, first select the picture to be detected, and then click the start detection button. The picture will be submitted to the function to detect the picture through Ajax. The function detects the picture. After the detection. The detected picture and count results are saved locally, while the path to save the file is returned in JSON format. Ajax receives the returned JSON file, the detected images and count results are displayed on the page. Fig. 2 shows the visual interface for picture detection.

Figure 2. Picture detection visualization

In Fig. 2, the right-hand side is the detected image, and the left-hand text is the detected objects and number of objects. For example, there are three people in the detected picture, the text description shows the three people synchronously.

B. *Video detection visualization*

Local video detection and online video detection are visualized in the same way. For the visualization of video detection, the video streaming method would be used. The video to be detected is submitted to the video stream function, which calls the video detection function to detect the incoming data. The detected data is output in the format of a video stream, the video stream is displayed on the web. At the same time, the detected target types and quantities are displayed in real time. The video detection visualization is shown in Fig. 3.

Figure 3. Video detection visualization

In Fig. 3, the video detection results and text descriptions are displayed in the same location as in Fig. 2. However, in Fig. 3, the video detection results and text descriptions are updated in real time.

C. *Text Visualization*

The display of count results for picture detection is a static operation, the detection results are displayed on the front page via Ajax transfer. The display of count results for video detection is a dynamic operation. This must record the type and number of objects for each frame, and display the results on the front page in real time. To ensure that the count results are updated in real time on the page, the setTimeout method in JS is used to partially update the part of the webpage that displays the counting results.

VI. CONCLUSION

The paper presents the SSD model, which is modified by adding the BN algorithm. The modified SSD model was trained and tested on Pascal VOC dataset and obtained a high mAP. There's not much of a drop in detection speed. The object counting function is added to the object detection. The useable object detection system was built using the Flask framework and Layui framework. The following will further improve the model by introducing Convolutional Block Attention Module (CBAM) [13]. This can improve the learning and generalization capabilities of the model, further improve the performance of SSD algorithms.

REFERENCES

[1] Girshick R B, Donahue J, Darrell T, et al. Rich feature hierarchies for accurate object detection and semantic segmentation[J]. IEEE Transactions on Pattern Analysis and Machine Intelligence. 2014:580-587.

[2] HE K M, ZHANG X Y, REN S, et al. Spatial pyramid pooling in deep convolutional networks for visual recognition[J]. IEEE Transactions on Pattern Analysis and Ma-chine Intelligence, 2015, 37 (9):1904-1916.

[3] Girshick R B. Fast R-CNN[C]. International Conference on Computer Vision. 2015:1440-1448.

[4] REN S, HE K, Girshick R, et al. Faster R-CNN: towards real-time object detection with region proposal networks [J]. IEEE Transactions on Pattern Analysis & Machine Intelligence, 2017, 39(6): 1137-1149.

[5] DAI J F, LI Y, HE K M, et al. R-FCN: Object detection via region-based fully convolutional networks[C]. NIPS. 2016:379-387.

[6] Redmon J, Divvala S, Girshick R, et al. You only look once: Unified, real-time object detection[C]. IEEE Conference on Computer Vision and Pattern Recognition. 2016:779-788.

[7] Liu W, Anguelov D, Erhan D, et al. SSD: Single shot multibox detector[C]. European Conference on Computer Vision, 2016:21-37.

[8] Jinchen Hu, Yuchen Wang, Jianghong Jiang, et al. Deep convolutional network based target detection Overview of measurement techniques [J]. Digital technology and applications. 2018, 36(4):97-98.

[9] FU C Y, LIU W, RANGA A, et al. DSSD: Deconvolutional single shot detector[J]. 2017: arXiv: 1701.06659.

[10] Simonyan K, Zisserman A. Very deep convolutional networks for large-scale image recognition[C]. ICLR, 2015.

[11] HE K M, ZHANG X Y, REN S, et al. Deep residual learning for image recognition[C]. CVPR. 2016:770-778.

[12] JEONG J, PARK H, KWAK N. Enhancement of SSD by concatenating feature maps for object detection[J]. Computer Vision and Pattern Recognition. 2017: arXiv: 1705.09587.

[13] Sanghyun Woo, Jongchan Park, Joon-Young Lee, et al. CBAM: Convolutional Block Attention Module[C]. ECCV. 2018: arXiv: 1807.06521.